

E5934-01EY
ASA-1000

Title of the Invention

MULTI-PROCESS DISPLAY METHOD
IN DEBUGGER SYSTEM

Inventors

Toshihiro TSURUGASAKI

- 1 -

MULTI-PROCESS DISPLAY METHOD IN DEBUGGER SYSTEM

BACKGROUND OF THE INVENTION

The present invention relates to a multi-process display method of displaying multiple processes to be debugged in a debugger system, a debugger system, 5 and a storage medium having stored thereon a display program, and in particular, to a method of and a system for debugging a multi-process program in which changes in operation states of processes of a multi-process program are displayed in relation to operation of the 10 program to thereby support the debugging.

Various methods have been devised to support debugging of a program. For example, JP-A-5-233323 describes a method to interactively and efficiently debug a parallel program. JP-A-8-185340 describes a 15 parallel program visualizing method as a tool to visualize a parallel program. However, the techniques of the prior art does not have a function to clearly and visually display such relationships between processes of a parallel program as a parent-child 20 relationship and a brother relationship. None of the techniques has a function to clearly display operation states of processes such as "generation", "start", "resume", "halt", and "end". JP-A-11-161514 describes a debugging method of a program including a plurality 25 of processes, but has not a function to debug processes

SUMMARY OF THE INVENTION

The object according to the present invention can be achieved by paying attention to a point in which debug information of a relationship between operation states of a plurality of processes of a multi-process program is clearly displayed in relation to operation of the program and a point in which a relationship between processes in operation and processes already ended is visually and discriminatively displayed. Resultantly, it is possible to provided a debug support method and a debug support system capable of efficiently debugging a multi-process program including a plurality of processes.

20 To achieve the object according to the
present invention, there is provided a multi-process
display method of displaying processes to be debugged
in the debugger system in which the debugger system
acquires an operation state change and detailed
25 information of each process, displays each process
using a predetermined pattern, displays a relationship
between processes using a layout of the patterns and

displayed on a display; and

Fig. 9 is a diagram showing a procedure to display a process graph and process detail information in an emphasized mode.

5 DESCRIPTION OF THE EMBODIMENTS

Referring now to the accompanying drawings, description will be given of an embodiment according to the present invention.

Fig. 1 shows a configuration of a debugger
10 system in a block diagram. The configuration includes a program 100 to be debugged (to be referred to as a debug program herebelow). The debugger system includes a debugger system section 110 and a display 150. The debug program 100 includes a plurality of processes (to
15 be referred to as debug processes herebelow) constituting the program 100. The processes are application programs prepared by users. When the user compiles a source program or code with a specification of "debug option", processes are generated in a format
20 suitable for the debugger system. That is, in each process, when an operation state is changed or when a particular system call is issued, an interruption signal is created to output information related to the debugging.

25 The debugger system section 110 includes a debugger engine section 120 which controls execution of the debug program 100 and which monitors the execution

thereof and a debugger information display section 140 to display an operation state of the program 100 on the display 150.

The debugger system section 110 is activated
5 by a command from a user. After activating the section 110, the user inputs to the section 110 a command or the like to determine a break point (at which execution of a program is temporarily stopped) for the debug
10 debug program. The debug program stops execution at a predetermined break point and a particular system call and passes control to the debugger system section 110. The user checks an operation state of each process and values of data and variables by commands. Thereafter,
15 when the user issues an instruction to resume a process, the execution of the debugger program is resumed. In this way, the debugger system section 110 provides an interface for the user to interactively execute the debug program.

20 The debugger engine section 120 includes a debug execution processing section 121 which controls execution of the debug program 100 for each debug process 101 and which monitors the execution and a process operation information output section 122. When
25 an operation state of the debug process 101 changes, the debug execution processing section 121 detects an interruption signal from the debug process 101 and transfers control to the process operation information

output section 122. The output section 122 outputs process operation information 131 to a process operation information file 130. Details of the information 131 will be described later by referring to

5 Fig. 2. A process operation information monitoring section 141 makes a check to determine presence or absence of the output of the process operation information 131 to the file 130 to change a display state of the display 150 in relation to operation of

10 the debug process 101. If the information 131 is outputted to the file 130, the monitoring section 141 passes control to a process operation information acquiring section 142. The section 142 receives the process operation information 131 from the file 130 and

15 then passes control to a process information table creating section 143. Using the process operation information 131, the section 143 creates and updates a process information table to control the debug process 101 and passes control to a process information display

20 section 144. Details of the process information table will be described later by referring to Fig. 3. The process information display section 144 includes a process graph display section 145 and a process detail information display section 146. The section 144

25 refers to the process information table created and updated by the table creating section 143 and displays a process graph 151 and process detail information 152 on the display 150.

The debugger engine section 120 and the debugger information display section 140 of the debugger system 110 are respectively software programs. Each part of the sections 120 and 140 is implemented using a subroutine. The process operation information file 130 and the process information table are respectively tables in a main memory. The implementation above is only an example, that is, the sections 120 and 140, the file 130, and the process information table can be implemented in other than this way.

Fig. 2 shows a data layout of the process operation information 131.

The process operation information file 130 to store the process operation information 131 has a queue configuration of first-in-first-out (FIFO) type. The process operation information output section 122 enqueues in the process operation information file 130 the process operation information 131 in either one of the layouts 210, 220, 230, 240, and 250 shown in Fig. 2. The operation information monitoring section 141 monitors the information and the process operation information acquiring section 142 dequeues the information from the file 130. The process operation information 131 includes process state information 201 and process detail information 202. The information 201 indicates an operation state of the debug process 101. The information 202 is disposed to control

information associated with the process state
information such as a number (to be referred to as a
node number) sequentially assigned to the debug process
101 by the debugger engine section 120, a process
5 number (assigned to the debug process 101 by an
operating system (OS)), a program name, and source file
information at a halt position. The process operation
information 210 is an example of the process operation
information 131 when a new debug process 101 is created
10 (corresponding to a system call "fork"). This means
that a debug process with node number "1" 214 and
process number "20" 215 is created from a debug process
with node number "0" 212 and process number "10" 213.
The process operation information 220 is an example of
15 the process operation information 131 when operation of
the process is executed (corresponding to a system call
"exec"). This means that execution of program name
"Prog2_0" 224 is started in a debug process with node
number "1" 222 and process number "20" 223. The
20 process operation information 230 is an example of the
process operation information 131 when operation of the
process is halted, for example, by a break point set by
the user. This means that a debug process with node
number "2" 232 and process number "21" 233 is halted at
25 line number "146" 235 of a source file "function.c"
234. The process operation information 240 is an
example of the process operation information 131 when
the operation of the debug process is resumed. This

means that the operation of a debug process with node number "0" 242 and process number "10" 243 is resumed. The process operation information 250 is an output example of the process operation information 131 when
5 the operation of the debug process is ended. This means that the operation of a debug process with node number "3" 252 and process number "30" 253 is ended.

Fig. 3 shows details of the process information table 300 which is created and updated by
10 the process information table creating section 143 and which is referred to by the process graph display section 145 and the process detail information display section 146. One process information table is created for each of the debug processes created up to the
15 pertinent point time. The information table 300 of each process includes debug process information 310, process box coordinate position information 340, child debug process information 320, and brother debug process information 330. The debug process information
20 310 includes node number information 311, process number information 312, program name information 313, and operation state information 314. The process box coordinate position information 340 includes information of coordinates when a debug process managed
25 by the process information table 300 is displayed as a process box in a process graph, that is, x-axis information 341, y-axis information 342, width information 343, and height information 344 of the

process box. The child debug process information 320 is a process information table pointer 321 to a debug process (to be referred to as a child process herebelow) created by the debug process managed by the process information table 300. The brother debug process information 330 is a process information table pointer 331 to a debug process (to be referred to as a brother process herebelow) which is a brother of the debug process managed by the process information table 300.

Fig. 4 is a flowchart of a processing procedure of the debugger information display section 140 to display on the display 150 the process operation information 131 received from the process operation information file 130. First, the process operation information monitoring section 141 periodically makes a check to determine whether or not the process operation information 131 is outputted to the file 130 (step 411). If the information 131 is outputted, the monitoring section 141 passes control to the process operation information acquiring section 142. The section 142 dequeues the information 131 from the file 130 (step 421) and passes control to the process information table creating section 143. The section 143 obtains an operation state of the debug process from the process state information 201 of the process operation information 131 (step 431). When the information 201 indicates a debug process generation

state, a process information table is created (step 432). When the information 201 indicates a debug process start, resume, halt, or end state, the process information table is updated (step 433). Control is

5 the passed to the process information display section 144. Fig. 5 shows an embodiment of the process information generation processing 432 in detail. Fig. 6 shows details of the process information update processing 433. Using the process information table,

10 the process information display section 144 repeatedly executes processing 442 to create data to display a process graph in the process graph display section 145 and processing 443 to create data to display process detail information in the process detail information

15 display section 146 as many times as debug process number (steps 441 to 444). The process information display section 144 resultantly display the process graph and the process detail information on the display 150. Embodiments of the process graph display data

20 generation processing 442 and the process detail information display data generation processing 443 will be specifically described later by referring to Fig. 7. After the graph and the information are displayed, steps 411 to 444 are repetitiously executed.

25 Fig. 5 shows a specific embodiment of the process information table generation processing 432. Description will be given of a case in which the process operation information 210 described by

referring to Fig. 2 is inputted to the processing 432. The program first retrieves a process information table managing a debug process (a debug process with node number "0" 212 and process number "10" 213 in this example and will be referred to as a parent process herebelow) having created a new debug process 101 (step 501). The retrieval is carried out by determining whether or not the node number (node number "0" 212) of the parent process matches node number information in the process information table (step 502). When the process information table managing the parent process is retrieved, the program creates a process information table of the new debug process (a debug process with node number "1" 214 and process number "20" 214 in this example and will be referred to as a child process herebelow; step 503) and registers to the child debug process information in the process information table of the parent process. Simultaneously, the program also registers brother debug process information and then passes control to the process information display section 144.

Fig. 6 shows a concrete embodiment of the process information table update processing 433. Description will be given of a case in which the process operation information 230 is inputted to the update processing 433 when the operation is halted, for example, at a break point shown in Fig. 2. First, the program retrieves a process information table managing

the debug process of which the operation is halted (the debug process with node number "2" 232 and process number "21" 233 in this example; step 601). The retrieval is carried out by determining whether or not the node number (node number "2" 233) of the debug process matches node number information in the process information table (step 602). When the process information table managing the debug process is retrieved, the program determines an operation state of the debug process according to the process state information 231 of the process operation information 230 (step 603). Since the process state information 231 of the process operation information 230 indicates "operation halt" in this case, the program sets operation halt information to the operation state information of the process information table managing the debug process (step 606) and passes control to the process information display section 144. Similarly, when the process operation information 220 is inputted, the program sets operation start information to the operation state information (step 604). When the process operation information 240 is inputted, the program sets operation resume information to the operation state information (step 605). When the process operation information 250 is inputted, the program sets operation end information to the operation state information (step 607). The program then passes control to the process information display section 144.

Fig. 7 is a specific embodiment of the process information display section 144. The program first reads a process information table created and updated by the process information table creating section 143 (step 701) and makes a check to determine whether or not the debug information display processing has been completely executed for all debug processes (step 702). If the processing has been completely executed, the program passes control to the display section 150. Since the process graph display section 145 displays a debug process in the form of a process box, the process information display section 144 calculates process box display position information (x-axis, y-axis, width, and height information; step 703) and sets the values resultant from the calculation to the process box coordinate position information of the process information table (step 704). The process box is fixed in size in principle. The display section 145 arranges the process boxes in a screen memory with a fixed interval therebetween such that processes having a parent-child relationship are horizontally disposed and processes of brothers are vertically disposed. To display the process boxes in relation to operation states (generation, start, resume, halt, and end) of the respective debug processes, the display section 145 refers to the operation state information of the process information table (step 705), creates process box display screen data according to the operation

state (step 710, 711, 712, 713, or 714), and then passes control to the process detail information display section 146. The display section 146 creates screen data to display process detail information (step 5 443).

Fig. 8 shows a display example of a process graph 151 and process detail information 152 displayed on one screen of the display 150. For the process boxes 801 to 804 displayed in the process graph 151, 10 debug process information, i.e., a process number (node number) of debug process information is displayed as information to indicate each debug process. For example, debug process information 810 of a process box 801 indicates that the debug process has process number 15 "10" (node number "0"). Since the process box 801 is linked by a line 811 with the process box 802, the process box 801 is a parent process having generated a debug process indicated by the process box 802. The process box 801 is also linked by a line 812 with the 20 process box 804, and hence the process box 801 is a parent process having generated a debug process indicated by the process box 804. It is accordingly to be appreciated that the process boxes 802 and 804 are bother processes. The process box 802 is linked by a 25 line 813 with the process box 803 and hence is a parent process having generated the debug process indicated by the process box 803. Neither of the process boxes 803 and 804 is linked rightward by a line with another

process box. Therefore, neither thereof has generated a child process. The debug process of the process box 801 is displayed with bold lines (841). This indicates that the debug process is in the operation resume state. The processes respectively of the process boxes 802 and 804 are displayed with ordinary lines (842, 843). This indicates that the debug processes are in the operation halt state. The process of the process box 803 is displayed with dotted lines (843). This indicates that the debug process is in the execution end state. As above, the process boxes are displayed according to the operation states of the respective processes. The process detail information 152 is debug information which includes information of rows such as detailed operation information 821 to 824 of the debug processes and information of columns such as a node number 831, a program name 832, a process number 833, an operation state 834, and a halt location 835.

As shown in Fig. 8, a process graph 151 and a process detail information 152 can be independently moved in the vertical and horizontal directions. In the example of Fig. 8, the graph 151 and the information 152 are vertically arranged to respectively occupy an upper half and a lower half of the screen. However, they may be displayed in different windows such that either one thereof are displayed on the screen or both thereof are displayed in a partially overlapped state. There may be disposed another window

to input debug commands.

Fig. 9 shows a procedure to conduct an operation 910 for a debug state selection state (to be referred to as an emphasized display) of a particular debug process from a process graph 911 and a procedure to conduct an operation 920 for a debug information emphasized display of a particular debug process.

The flow shown in Fig. 9 is a part of the process information display section 144. The flow is activated by the operation of a mouse or a keyboard on the process graph 151 or the process detail information 152.

The emphasized display operation 910 from the process graph 911 is explained. When the debug information emphasized display operation 912 is carried out for a debug process with process number "20" (node number "1") using a mouse or the like from the process graph 911, the program retrieves, according to a point location of the mouse and coordinate information managed by the process box coordinate position information of the process information table, a process information table of a debug process displayed as a process box at the mouse point location (step 913). From the retrieved process information table, it is known that the debug process for which the emphasized display operation is conducted has a node number "1" (step 914). Using the node number as a key, a retrieval is conducted through the process detail

information rows displayed in the process detail
information 916 to detect information row having a node
number "1" (step 915). Process box emphasized display
processing (step 930) is executed using process box
5 coordinate position information of the process
information table retrieved by the process information
table retrieval (step 913). Emphasis display
processing of the process detail information is
executed according to information in the detected row
10 (step 931). Resultantly, a process box 943 indicating
node number "1" of a process graph 941 and process
detail information row 944 indicating the debug process
with node number "1" of process detail information 942
are emphasized in the presentation on a display 940 in
15 a related fashion. Next, an emphasized display
operation 920 from the process detail information 921
is explained. When a debug information emphasized
display operation 922 of the debug process with process
number "20" and node number "1" is conducted from the
20 process detail information 921 using, for example, a
mouse; a character string indicating, as row
information, detailed information of the debug process
is acquired using the mouse point location (step 923)
and a node number is obtained from the acquired
25 character string (step 924). Using the obtained node
number as a key, the program retrieves a process
information table managing the debug process for which
the emphasized display operation has been conducted

(step 925). As in the emphasized display operation 910 from the process graph, a process box 943 indicating the debug process with node number "1" of the process graph 941 and process detail information 944 indicating
5 the debug process with node number "1" of process detail information 942 are displayed on the display 940 in an emphasized mode in a related fashion.

In Fig. 9, a reverse display mode is adopted for the emphasized display. However, the width and the
10 color of the characters and the process boxes may be changed for this purpose.

The process graph 151 and the process detail information 152 are kept displayed until execution of all processes is finished and the user instructs the
15 debugger system to end its operation.

A variation of the embodiment will be next described.

In the embodiment, the process operation information output section 122 outputs the process
20 operation information 131 to the file operation information file 130. The process operation information monitoring section 141 monitors by polling the process operation information 131. In place of the processing, it may be conducted that using a task queue
25 or an inter-task communication, the process operation information output section 122 directly passes the process operation information 131 to the process operation information acquiring section 142 and

simultaneously triggers the start of the information
acquiring section 142.

The specification and drawings are,
accordingly, to be regarded in an illustrative rather
5 than a restrictive sense. It will, however, be evident
that various modifications and changes may be made
thereto without departing from the broader spirit and
scope of the invention as set forth in the claims.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213